An Exact Algorithm for a Vehicle Routing Problem with Time Windows and Multiple Use of Vehicles

Nabila Azi
Michel Gendreau
Jean-Yves Potvin

Département d'informatique et de recherche opérationnelle
and
Centre interuniversitaire de recherche sur les réseaux d'entreprise,
la logistique et le transport,
Université de Montréal,
C.P. 6128, succursale Centre-ville,
Montréal, Québec, Canada H3C 3J7.

Abstract

The vehicle routing problem with multiple use of vehicles is a variant of the classical vehicle routing problem. It arises when each vehicle performs several routes during the workday due to strict time limits on route duration (e.g., when perishable products are transported). The routes are defined over customers with a revenue, a demand and a time window. Given a fixed-size fleet of vehicles, it might not be possible to serve all customers. Thus, the customers must be chosen based on their profitability (i.e., revenues minus traveling costs). In this paper, a column generation approach is first proposed to address this problem. The master problem is a variant of a set covering problem, while the pricing subproblems are elementary shortest path problems with resource constraints. The column generation algorithm is then embedded within a branch-and-price framework to obtain integer solutions. Computational results are reported on Euclidean problems derived from well-known benchmark instances for the vehicle routing problem with time windows. Additional results are reported on highly constrained instances where the width of the time windows is reduced.

Keywords: Vehicle routing, time windows, multiple use of vehicles, elementary shortest paths with resource constraints, column generation, branch-and-price.

1 Introduction

We consider a variant of the Vehicle Routing Problem with Time Windows (VRPTW) where each vehicle can perform several routes during its workday. Surprisingly, this problem has received little attention in the literature in spite of its importance in

practice. For example, in the home delivery of perishable goods, like food, routes are of short duration and must be combined to form a complete workday. We believe that this type of problem will become increasingly important in the future with the advent of electronic services, like e-groceries, where customers can order goods through the Internet.

The vehicle routing problem with multiple use of vehicles, but no time windows, has been addressed through heuristic means in [9, 16]. In [16], different solutions to the classical vehicle routing problem are generated using a tabu search heuristic. The routes obtained are then combined to produce workdays for the vehicles by solving a bin packing problem, an idea previously introduced in [9]. A recent work in [5] describes insertion heuristics that can efficiently handle different types of constraints including time windows and multiple use of vehicles. Logistics and socio-economic considerations about different types of home delivery problems, with a particular emphasis on electronic groceries, can also be found in [10, 11, 12, 13, 17].

In [1], we proposed an exact algorithm for solving the single vehicle variant of the problem. This algorithm is a two-phase method in which all non-dominated feasible routes are first generated; some of these routes are then selected and sequenced to form the vehicle workday. In this paper, we extend this work to the much more challenging multiple vehicle case. To the best of our knowledge, this is the first time that an exact algorithm is devised for such a problem.

The outline of the paper is as follows. In Section 2, a mathematical programming formulation is proposed. The column generation scheme is presented in Section 3. Section 4 then describes the branch-and-price framework within which the column generation algorithm is applied at each node of the search tree. Computational results on problem instances derived from Solomon's VRPTW benchmarks [15] are reported in Section 5. Finally, concluding remarks follow in Section 6.

2 Problem Formulation

Our problem can be stated as follows. We have a fixed-size fleet of vehicles (each of capacity Q) denoted by set V, that delivers perishable goods from a depot to a set of customer nodes $N = \{1, 2, ..., n\}$ in a complete directed graph with arc set A. A distance d_{ij} and a travel time t_{ij} are associated with every arc $(i, j) \in A$. Each customer $i \in N$ is characterized by a revenue g_i, a demand q_i, a service or dwell time s_i and a time window $[a_i, b_i]$, where a_i is the earliest time to begin service and b_i is the latest time. Hence, a vehicle must wait if it arrives at customer i before a_i. The working day of each vehicle is made of a sequence of routes where each route starts and ends at the depot (some of these routes might be empty). These routes are denoted by set K, where $|K|$ is large enough to accommodate the maximal number of routes that the fleet can possibly perform in a day. We assume, without loss of generality, that the routes served by any vehicle are numbered in increasing order, that is, a vehicle serves route l after route k if and only if $l > k$.

The depot is denoted by 0 or $n+1$ depending if it is the initial or terminal node

of an arc, with $s_0 = s_{n+1} = 0$, $q_0 = q_{n+1} = 0$, $a_0 = a_{n+1} = 0$; $b_0 = b_{n+1} = \infty$; the symbol N^+ is used for $N \bigcup \{0, n+1\}$ and A^+ for $A \bigcup \{(0, n+1)\}$, where $(0, n+1)$ is a dummy arc with distance $d_{0,n+1} = 0$ and travel time $t_{0,n+1} = 0$. Every customer in a route must be served before a given deadline associated with that route. The latter is defined by adding a constant t_{max} to the route start time. Also, a setup time σ^k for loading the vehicle is associated with each route $k \in K$.

In practice, it might not be profitable nor feasible, due to the time window constraints, to serve all customers with the fleet of vehicles. The objective that we consider is thus to minimize the difference between the total distance traveled and the total revenue earned from served customers (weighted by a parameter α).

The following variables are used in the formulation of the problem.

- For each route $k \in K$ and each arc $(i, j) \in A^+$, binary variable x_{ij}^k indicates whether or not arc (i, j) appears in route k. (Note that when $x_{0,(n+1)}^k = 1$, route k is empty.)
- For each route $k \in K$ and each customer $i \in N$, binary variable y_i^k indicates whether or not customer i is served by route k.
- For each route $k \in K$ and each customer $i \in N$, continuous variable t_i^k indicates when service starts at customer i if it is served by route k. When customer i is not served by route k, this value is meaningless. For each route $k \in K$, t_0^k (resp. t_{n+1}^k) is the time at which the route starts (resp. ends) at the depot.
- For each pair of routes $k, l \in K$ with $k < l$, binary variable z_{kl} indicates whether or not route l immediately follows route k in the workday of one of the vehicles.

This problem can be formulated as follows, with M an arbitrary large constant:

$$\text{Min} \sum_{k \in K} \sum_{(i,j) \in A} d_{ij} x_{ij}^k - \alpha \sum_{k \in K} \sum_{i \in N} g_i y_i^k \tag{1}$$

subject to

$$\sum_{j \in N^+} x_{ij}^k = y_i^k, \quad i \in N,\ k \in K, \tag{2}$$

$$\sum_{k \in K} y_i^k \leq 1, \quad i \in N, \tag{3}$$

$$\sum_{i \in N^+} x_{ih}^k - \sum_{j \in N^+} x_{hj}^k = 0, \quad h \in N,\ k \in K, \tag{4}$$

$$\sum_{i \in N^+} x_{0i}^k = 1, \quad k \in K, \tag{5}$$

$$\sum_{i \in N^+} x_{i(n+1)}^k = 1, \quad k \in K, \tag{6}$$

$$\sum_{i \in N} q_i y_i^k \leq Q, \quad k \in K, \tag{7}$$

$$t_i^k + s_i + t_{ij} - M\left(1 - x_{ij}^k\right) \leq t_j^k, \quad (i,j) \in A^+, \ k \in K, \tag{8}$$

$$a_i y_i^k \leq t_i^k \leq b_i y_i^k, \quad i \in N, \ k \in K, \tag{9}$$

$$t_0^k \geq \sigma^k, k \in K, \tag{10}$$

$$t_i^k \leq t_0^k + t_{max}, \quad i \in N, \ r \in K, \tag{11}$$

$$\sigma^k = \beta \sum_{i \in N} s_i y_i^k, \quad k \in K, \tag{12}$$

$$t_0^l + M(1 - z_{kl}) \geq t_{n+1}^k + \sigma_l, \quad k,l \in K \ k < l, \tag{13}$$

$$\sum_{k \in K} \sum_{l \in K | l > k} z_{kl} \geq |K| - |V| \quad k,l \in K, \tag{14}$$

$$x_{ij}^k \in \{0,1\}, \quad i,j \in A, \ k \in K, \tag{15}$$

$$y_i^k \in \{0,1\} \quad i \in N, \ k \in K, \tag{16}$$

$$z_{kl} \in \{0,1\} \quad k,l \in K, \ k < l, \tag{17}$$

$$t_i^k \geq 0, \quad i \in N, \ k \in K. \tag{18}$$

In this formulation, equations (2) and (3) state that every customer should be visited at most once. Equations (4), (5), and (6) are flow conservation constraints that describe the individual routes. Equation (7) states that the total demand on a route cannot exceed vehicle capacity. Equations (8), (9), (10) and (11) ensure feasibility of the time schedule. Note that equation (11) corresponds to the deadline constraint for the service at a customer and that equation (9) forces the t_i^k variables to 0 when customer i is not in route k. Consequently, equation (11) is automatically satisfied in this case. Equation (12) defines the vehicle loading time for a route as the sum of the service times of all customers in that route multiplied by a constant β. Constraints (13) and (14) ensure the proper route sequencing within the workdays of individual vehicles. Note that through equation (14), at most $|V|$ routes are at the beginning of workdays, which implies that there are at most $|V|$ workdays.

In the following, we assume that $g_i = 1$, for all $i \in N$. Also, we set α to a large value, so that the inclusion of any additional customer is always profitable. In this way, we obtain a hierarchical objective where the number of served customers is first maximized and, for the same number of served customers, the total travel distance is minimized.

3 Column generation

In practice, the formulation in Section 2 is unlikely to be tractable for any instance of reasonable size. We thus propose to address this problem with a column-generation approach embedded within a branch-and-price framework. Column generation is well documented in the literature. We will thus only briefly introduce the master problem and the pricing subproblem in the following. The interested reader will find more details about column generation in [2, 7].

3.1 Master problem

In the master problem (MP), every column corresponds to a workday r. The decision variables x_r are binary variables that indicate if each workday r is used or not. The MP can then be written as:

$$\text{Min} \quad \sum_{r\in\Omega}(d_r - \alpha g_r)x_r \tag{19}$$

$$\text{s. t.} \quad \sum_{r\in\Omega} a_{ir}x_r \leq 1, \quad i \in N, \tag{20}$$

$$\sum_{r\in\Omega} x_r \leq |V|, \tag{21}$$

$$x_r \in \{0,1\}, \quad r \in \Omega. \tag{22}$$

where Ω is the set of all feasible workdays, g_r is the number of customers in workday r (recall that $g_i = 1, i \in N$), d_r is the total travel distance in workday r and a_{ir} is 1 if customer i is in workday r, 0 otherwise. A solution is thus a subset of workdays Ω' taken from Ω that covers each customer at most once.

As the number of columns can be huge, the columns are progressively introduced into the master problem to obtain a number of restricted MPs. The linear relaxation of these restricted MPs (RLMPs) is then solved with CPLEX. The dual variables associated with the optimal solution of a given RLMP are used to define a pricing subproblem, which is basically an elementary shortest path problem with resource constraints defined on an auxiliary graph. Solving this subproblem allows the identification of workdays with negative reduced cost, if any. The latter are then added to the current RLMP to obtain the next RLMP, and its linear relaxation is solved again to obtain new dual variables. This iterative procedure is repeated until no more paths with negative reduced costs can be found. At this point, an optimal solution for the linear relaxation of the MP has been obtained.

3.2 Pricing subproblem

The pricing subproblem consists in finding an elementary shortest path with resource constraints on an auxiliary graph $G^T = (A^T, N^T)$. The latter, called the route

graph, is constructed as follows. First, the algorithm proposed by Feillet et al. [8] is applied on the original graph to find all non dominated feasible routes. These routes, plus two artificial nodes that correspond to the start and end of the workday, form node set N^T. The arc set A^T corresponds to feasible transitions between routes. In particular, given two routes k and l, arc $(k,l) \in A^T$ if routes k and l have no customers in common and route l can be feasibly served after route k (where feasibility relates to the time window constraints). The arc cost c_{kl} corresponds to $d_l - \alpha g_l$, where d_l is the distance of route l and g_l is the number of customers in route l. There is also an arc from the artificial start node to every route node and from every route node to the artificial end node (in the latter case, the costs are set to 0). Note that this route graph is constructed once and for all at the beginning of the algorithm. More details about the construction of this graph can be found in [1].

We can now formulate the pricing subproblem using binary variables X_{kl} which indicate if arc (k,l) in used in the route graph. The reduced cost of arc (k,l) is denoted $c'_{kl} = c_{kl} - \sum_{i \in N_l} \lambda_i$, where N_l is the set of customers in route l and λ_i is the dual variable associated with constraint (20) for customer i, and μ is the dual variable associated with constraint (21).

$$\text{Min} \sum_{(k,l) \in A^T} c'_{kl} X_{kl} - \mu \tag{23}$$

$$\text{s. t.} \quad \sum_{(k,h) \in A^T} X_{kh} - \sum_{(h,l) \in A^T} X_{hl} = 0, \quad h \in N^T, \tag{24}$$

$$\sum_{k \in N^T} X_{0k} = 1, \tag{25}$$

$$\sum_{k \in N^T} X_{k(n+1)} = 1, \tag{26}$$

$$T_k + \sigma_l + \left(\bar{t}^k_{n+1} - \bar{t}^k_0\right) - M(1 - X_{kl}) \leq T_l, \quad (k,l) \in A^T, \tag{27}$$

$$\underline{t}^k_0 \leq T_k \leq \bar{t}^k_0, \quad k \in N^T, \tag{28}$$

$$X_{kl} \in \{0,1\}, \quad (k,l) \in A^T, \tag{29}$$

$$T_k \geq 0, \quad k \in N^T. \tag{30}$$

In this formulation, nodes 0 and $n+1$ are the artificial start and end nodes, respectively. Continuous variable T_k corresponds to the vehicle departure time of route k while $\underline{t}^k_0$, $\bar{t}^k_0$, $\underline{t}^k_{n+1}$ and $\bar{t}^k_{n+1}$ are the earliest departure time, latest departure time, earliest return time, latest return time of route k, respectively. These bounds are derived from the time windows of the customers served in that route. Note also that $(\bar{t}^k_{n+1} - \bar{t}^k_0)$ corresponds to the route duration, which is also the minimum duration because the waiting time is minimized by serving the route at the latest feasible time.

Once again, we do not solve that formulation directly, but rather apply the

elementary shortest path algorithm with resource constraints of Feillet et al. [8] to find the best path from the artificial start node to the artificial end node in the route graph (i.e., to find a sequence of routes or workday). This algorithm is briefly introduced below.

3.3 Solving the pricing subproblem

The algorithm of Feillet et al., denoted FDGG in the following, is a label correcting algorithm that solves the elementary shortest path problem with resource constraints. In this context, a path is characterized by the consumption of each resource, in addition to its cost. Accordingly, when different paths lead to the same node, it might well be that no path dominates, or is better than the others, over all criteria. As a consequence, many different labels are typically maintained at each node (i.e., all non-dominated paths leading to that node).

Since elementary paths must be generated, cycles are detected by keeping a trace of previously visited nodes. More precisely, a path p from some origin node o to some node $j \in N$ is labeled with $R_p = (c_p, t_p^1, \ldots, t_p^l, s_p, V_p^1, \ldots, V_p^n)$, where n is the number of nodes in the graph, $L = \{1, ..., l\}$ is the set of resources, c_p is the cost of path p, t_p^k is the consumption of resource $k = 1, ..., l$, s_p is the number of unreachable nodes (either because they have already been visited or because their inclusion would violate one or more resource constraints) and $V_p^i = 1$ if node i is unreachable, 0 otherwise. The following dominance relation is then defined:

Dominance relation. If p and p' are two different paths from origin o to node j with labels R_p and R'_p, respectively, then path p dominates p' if and only if $c_p \leq c_{p'}$, $s_p \leq s_{p'}$, $t_p^k \leq t_{p'}^k$, $k = 1, ..., l$, $V_p^i \leq V_{p'}^i$, $i = 1, ..., n$.

That is, path p dominates p' if (1) it is not longer, (2) it does not consume more resources for every resource considered and (3) every unreachable node is also unreachable for path p'. Note that s_p, the number of unreachable nodes, is included in the label only to speed up the computations. As stated in [8], by eliminating paths through this dominance relation, only labels corresponding to non dominated elementary paths are kept and a solution to the problem is obtained at the end.

The FDGG algorithm is used to find elementary least cost paths with time window constraints from the artificial start node to the artificial end node in the route graph. The time windows for vehicle departure associated with each route are used for this purpose. That is, the vehicle must be back at the depot and ready to depart before the latest departure time of its next route. When arc (k, l) is added to the current path, route l is added to the vehicle workday. In this case, the cost of route l is incurred and the time consumed corresponds to the duration of route l, plus the setup time of route l, plus any waiting time before departure. The sequence of nodes in a least cost path obtained at the end corresponds to the sequence of routes in the vehicle workday.

To speed up the computations, we start with a heuristic version of this algorithm where the condition $V_p^i \leq V_{p'}^i$, $i = 1, .., n$, is not checked in the dominance relation between paths p and p' (i.e., path p' is eliminated even if it is not really dominated

by p). The exact algorithm is used only when the heuristic cannot find a workday of negative reduced cost.

3.4 The column generation algorithm

In this subsection, we explain how the column generation algorithm is initialized, how the columns are managed, and what are the lower bounding and termination criteria. This algorithm is executed at each node of the search tree.

Initialization. At the root of the search tree, the RLMP is initialized with workdays made of a single customer. The number of columns thus corresponds to the number of customers. For the other nodes in the search tree, the algorithm initializes the RLMP with the set of columns in the last node considered, after removing columns that are infeasible due to branching (see Section 4).

Column management. We stop solving the pricing subproblem when a large number of columns with negative reduced costs has been found. This number is set to 200 in our experiments. That is, we do not want to spend too much time in the pricing subproblem, given that a column with an optimal reduced cost value is not really needed.

Lower bounding and termination. One disadvantage of the column generation algorithm is that a lot of iterations are often needed to prove optimality, while the optimal value of the RLMP does not change significantly from one iteration to the next. Many authors have thus proposed alternative lower bounding techniques that allow earlier termination. In particular, the following dual bound $\theta(\lambda)$ can be derived from the lagrangean relaxation of constraints (20), where λ stands for the vector of dual variables associated with these constraints:

$$\theta(\lambda) = |V|c'^* + Z^*_{RLMP} \ .$$

In this formula, Z^*_{RLMP} is the optimal value of the current RLMP and c'^* is the optimal reduced cost of the subproblem, which depends on λ. If this dual bound is greater than or equal to the best incumbent feasible solution (primal bound), the column generation algorithm is stopped.

4 The branch-and-price algorithm

The column generation algorithm described in Section 3 is run at every node of a search tree within a branch-and-price framework. This section thus describes the search and branching strategies that we used.

4.1 Search strategy

The search tree is explored according to a best-first policy, where subproblems are ranked according to their associated lower bound. Best-first was chosen over a depth-

first search policy, as the latter yielded inferior results in preliminary experiments.

4.2 Branching

Different branching schemes are applied in the following order: branching on the number of vehicles, branching on customers and branching on arcs. All these strategies are compatible with the structure of the pricing subproblem. They are explained in the following.

Branching on the number of vehicles. First, we determine the number of vehicles $v = \sum_{r \in \Omega'} x_r$ in the optimal solution of the current relaxed RLMP. If this number is fractional, we impose $\sum_{r \in \Omega'} x_r \leq \lfloor v \rfloor$ in the first branch and $\sum_{r \in \Omega'} x_r \geq \lfloor v + 1 \rfloor$ in the second branch. This branching rule does not impact the subproblem.

Branching on customers. We search for a customer i for which $y_i = \sum_{r \in \Omega'} a_{ir} x_r$ is fractional. If several fractional customers exist, the one with a value closest to 0.5 is selected. Two branches are created which respectively forbid ($y_i = 0$) and enforce ($y_i = 1$) customer i in the solution. In the first case, all columns or workdays that serve customer i are deleted from the RLMP, as well as all routes with this customer in the route graph. In the second case, the corresponding constraint in (20) is forced to 1.

Branching on customers can make the RLMP infeasible because, once a substantial number of branching decisions has been made, many constraints of type (20) can be forced to one. This situation is addressed by adding artificial z_i variables in (20) to obtain $z_i + \sum_{r \in \Omega'} a_{ir} x_r = 1, i \in N$. A very large cost is assigned to these variables to make sure that they are part of the solution only in case of infeasibility.

Branching on arcs. We also branch when the flow on any arc (i, j) is fractional. Two branches are created: one branch with $x_{ij} = 1$, where vertex j must be visited immediately after vertex i; and the other branch with $x_{ij} = 0$, where it is forbidden to visit vertex j after vertex i. In the first case, all workdays in the RLMP, as well as all routes in the route graph, that contain arc (i, k) with $k \neq j$ or arc (k, j) with $k \neq i$ are deleted. Also, vertices i and j are forced in the RLMP by setting $y_i = 1$ and $y_j = 1$. In the second case, all workdays in the RLMP, as well as all routes in the route graph, where vertex j immediately follows vertex i, are deleted.

We also branch on two consecutive fractional arcs at once to reach integrality faster. Assuming that x_{ij} and x_{jk} are both fractional and close to 0.5, we impose $x_{ij} + x_{jk} \geq 1$ on one branch and $x_{ij} + x_{jk} = 0$ on the other branch. The RLMP and the route graph are then updated accordingly. We also generalized this concept to more than two arcs, but no significant improvement was observed.

5 Computational Results

Solomon's classical VRPTWs [15] were used for this study. In these problems both the distance and travel time between two customer locations correspond to the Euclidean distance. There are six different classes of instances depending on the

geographic location of the customers (R: random; C: clustered; RC: mixed) and width of scheduling horizon (1: short horizon; 2: long horizon). In this study, instances of type 1 have been discarded due to the short horizon that does not allow a significant number of routes to be sequenced to form a workday. Results are thus reported for R2 (11 instances), C2 (8 instances) and RC2 (8 instances). All tests were run on an AMD Opteron 3.1 MHz with 16 GB of RAM, using ILOG CPLEX 10.0 to solve the RLMPs. The time limit for each run of the branch-and-price algorithm was set to 25 hours.

Solomon's VRPTWs were modified to fit our problem. In particular, a deadline constraint t_{max} was associated with each route. Based on the results obtained on the single vehicle variant of the problem [1], this parameter was set to 75 in the cases of R2 and RC2, and to 220 in the case of C2. This value needs to be larger for C2 because the service time at each customer is 90, as opposed to 10 for R2 and RC2. It should also be noted that the algorithm is very sensitive to the t_{max} value and that the route graph quickly becomes unmanageable when larger values are used. Finally, parameter β for the route loading time in equation (12) was set to 0.2 in all experiments.

For each instance, only the first 25 or 40 customers, out of 100 customers, were considered. The results were recorded at the root node and at the end of the branch-and-price algorithm (due either to timeout or optimality). These results are shown in Tables 1 to 5. In these tables, a particular instance is identified by its class and its index followed by a dot and the number of customers considered. For example RC202.40 is the second instance of class RC2, where only the first 40 customers are considered.

In the tables, column *Problem* is the problem identifier, *Gap* is the gap in % between the value of the linear relaxation at the root and the optimal value, *Cols* is the total number of columns generated during the branch-and-price algorithm, *Iter* is the total number of RLMPs that were solved by CPLEX, *Nodes* is the number of nodes explored in the search tree, *Dist.* is the total travel distance, *% Cust.* is the percentage of served customers, *Routes per day* is the average number of routes in a workday, *Cust. per route* is the average number of customers in a route and *CPU* is the computation time in seconds.

Tables 1 to 3 show that our algorithm can solve all instances with 25 customers and 2 vehicles to optimality, with the exception of instances RC204.25, RC208.25 and C203.25, and that all customers are served by the vehicles. The CPU times vary widely and range from a few seconds to a few hours. When the number of customers is increased to 40, 11 problems out of 27 were solved to optimality, as indicated in Table 4, with a sharp increase in CPU time. In some cases it was not possible to serve all customers, even if the number of routes per workday significantly increased. Finally, we tried to help our algorithm by reducing in half the width of the time windows (i.e., by working on more constrained instances). The results for class RC2 are shown in Table 5. As we can see, we were able to solve three more instances in this class, with a spectacular reduction in CPU time. Only instance RC204.40, which seems to be particularly difficult, could not be solved within the alloted time.

It is worth noting that the problem is not harder to solve by our algorithm when the number of vehicles is increased, because the route graph remains the same. This is to be opposed to the number of customers and the value of the route deadline t_{max} that lead to a quick increase in the size of this graph (i.e., a quick increase in the number of feasible routes).

Problem	Gap	Cols	Iter	Nodes	Dist.	% Cust.	Routes per day	Cust. per route	CPU (sec.)
rc201.25	0.2%	1363	266	19	988.05	100%	6.0	2.1	988.1
rc202.25	3.1%	13496	670	67	881.49	100%	5.5	2.3	715.6
rc203.25	3.0%	10038	332	53	751.39	100%	4.5	2.8	753.7
rc205.25	2.2%	1693	194	21	840.35	100%	5	2.5	16.4
rc206.25	2.2%	7106	587	61	761.03	100%	4.5	2.8	138.1
rc207.25	6.1%	22377	1975	197	743.20	100%	4.5	2.8	4868.8

Table 1: RC2 instances with 25 customers and 2 vehicles

Problem	Gap	Cols	Iter	Nodes	Dist.	% Cust.	Routes per day	Cust. per route	CPU (sec.)
r201.25	0.4%	1787	230	9	762.43	100%	6.0	2.1	8.5
r202.25	0.0%	1500	25	1	645.78	100%	4.5	2.8	37.3
r203.25	0.1%	3074	43	3	621.97	100%	4.5	2.8	489.1
r204.25	0.3%	8226	159	23	581.50	100%	4.0	3.1	4772.0
r205.25	0.6%	6965	479	29	634.09	100%	4.5	2.8	209.3
r206.25	0.0%	1893	22	1	596.74	100%	4.0	3.1	143.2
r207.25	0.1%	8947	203	86	585.74	100%	4.0	3.1	3786.3
r208.25	0.3%	5052	95	91	579.68	100%	4.0	3.1	4769.3
r209.25	0.3%	3614	170	15	602.39	100%	4.0	3.1	252.0
r210.25	1.4%	9794	510	53	637.21	100%	4.5	2.8	1834.4
r211.25	0.5%	10514	331	97	575.91	100%	4.0	3.1	7734.0

Table 2: R2 instances with 25 customers and 2 vehicles

Problem	Gap	Cols	Iter	Nodes	Dist.	% Cust.	Routes per day	Cust. per route	CPU (sec.)
c201.25	0.9%	5873	1121	65	659.02	100%	5.5	2.3	302.9
c202.25	1.3%	27886	1237	115	653.37	100%	5.5	2.3	9000.5
c204.25	0.7%	38469	1271	233	602.59	100%	4.5	2.8	55881.8
c205.25	2.5%	15617	1880	103	641.93	100%	5.5	2.3	4519.5
c206.25	2.3%	30314	2708	167	636.39	100%	5.5	2.3	19926.5
c207.25	1.0%	48636	2109	159	603.22	100%	5.0	2.5	48013.3
c208.25	1.1%	15584	1430	111	613.20	100%	5.0	2.5	5793.8

Table 3: C2 instances with 25 customers and 2 vehicles

Problem	Gap	Cols	Iter	Nodes	Dist.	% Cust.	Routes per day	Cust. per route	CPU (sec.)
rc201.40	2.5%	3492	973	191	1372.25	80%	8.0	2.0	59.4
rc202.40	1.5%	18485	959	79	1431.62	92.5%	8.0	2.3	2150.4
rc203.40	1.4%	26825	1234	101	1468.44	100%	8.0	2.5	78913.2
rc205.40	2.4%	6496	1124	167	1330.73	87.5%	7.5	2.3	613.4
r201.40	0.2%	4291	494	13	1173.28	97.5%	9.0	2.2	535.2
r202.40	0.2%	10516	210	9	1042.09	100%	7.0	2.9	86493.5
r205.40	0.3%	12972	565	15	999.42	100%	7.0	2.9	23595.2
r206.40	0.0%	8673	74	3	918.61	100%	6.0	3.3	89461.8
c201.40	0.6%	17360	1662	47	1125.33	100%	8.5	2.4	2008.8
c205.40	0.9%	74560	6375	79	1091.50	100%	8.0	2.5	40179.6
c206.40	1.3%	46314	2866	85	1104.24	100%	8.5	2.4	24785.5
c208.40	0.8%	28990	848	17	1081.34	100%	8.0	2.5	5281.7

Table 4: RC2, R2 and C2 instances with 40 customers and 2 vehicles

Problem	Gap	Cols	Iter	Nodes	Dist.	% Cust.	Routes per day	Cust. per route	CPU (sec.)
rc201.40	0.0%	76	8	1	766.85	47.5%	4.5	2.1	0.0
rc202.40	0.0%	396	28	1	877.50	60%	5.0	2.4	2.3
rc203.40	4.1%	6828	1984	857	855.15	65%	5.0	2.6	3334.6
rc205.40	0.0%	207	16	1	843.61	50%	5.0	2.0	0.2
rc206.40	2.3%	371	67	21	851.08	55%	5.0	2.2	1.0
rc207.40	1.0%	783	96	7	834.13	65%	5.0	2.6	6.8
rc208.40	0.2%	841	63	11	850.28	70%	5.0	2.8	27.8

Table 5: RC2 constrained instances with 40 customers and 2 vehicles

6 Conclusion

In this paper, the first exact branch-and-price algorithm for solving the vehicle routing problem with time windows and multiple use of vehicles was proposed. An elementary shortest path algorithm was also exploited to solve the pricing subproblems. Although the algorithm is limited by problem size and some characteristics of the problem (like the time deadline of each route), it can still routinely solve problems with 25 customers and some problems with up to 40 customers. Clearly, a heuristic approach remains a viable alternative for larger instances and we are now focusing our current research efforts in that direction.

Acknowledgments. Financial support for this work was provided by the Canadian Natural Sciences and Engineering Research Council (NSERC). This support is gratefully acknowledged.

References

[1] Azi N., Gendreau M., Potvin J.-Y., "An Exact Algorithm for a Single Vehicle Routing Problem with Time Windows and Multiple Routes", *European Journal of Operational Research 178*, 755–766, 2007.

[2] Barnhart C., Johnson E., Nemhauser G., Savelsbergh M. Vance P., "Branch-and-price : column generation for solving huge integer programs", *Operations Research 46*, 316–329, 1998.

[3] Bent R., van Hentenryck P., "A Two-Stage Hybrid Local Search for The Vehicle Routing Problem with Time Windows", *Transportation Science 38*, 515–530, 2004.

[4] Campbell A.M., Savelsbergh M., "Decision Support for Consumer Direct Grocery Initiatives", *Transportation Science 39*, 313–327, 2005.

[5] Campbell A.M., Savelsbergh M., "Efficient Insertion Heuristics for Vehicle Routing ans Scheduling Problems", *Transportation Science 38*, 369–378, 2004.

[6] Czech Z.J., Czarnas, P., "A Parallel Simulated Annealing for the Vehicle Routing Problem with Time Windows", in *Proceedings of the 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing*, Canary Islands, Spain, 376–383, 2002.

[7] Desrosiers J., Dumas Y., Solomon M.M., Soumis F., "Time Constrained Routing and Scheduling", in *Network Routing*, M.O. Ball, T.L. Magnanti, C.L. Monma, G.L. Nemhauser (Eds.), North Holland, 35–139, 1995.

[8] Feillet D., Dejax P., Gendreau M., Gueguen C., "An Exact Algorithm for the Elementary Shortest Path Problem with Resource Constraints: Application to Some Vehicle Routing Problems", *Networks 44*, 216–229, 2004.

[9] Fleischmann B., "The Vehicle Routing Problem with Multiple Use of Vehicles", Working Paper, Fachbereich Wirtschaftswissenschaften, Universität Hamburg, Germany, 1990.

[10] Kilpala H.K., "The Impact of Electronic Commerce on Transport & Logistics in the Retail Grocery Industry", M.Sc. Thesis, University of Oulu, Finland, 1999.

[11] Lin I.L., Mahmassani H.S., "Can online grocers deliver? Some Logistics Considerations", *Transportation Research Record 1817*, 17–24, 2002.

[12] Punakivi M., Saranen J., "Identifying the Success Factors in E-Grocery Home Delivery", *International Journal of Retail and Distribution Management 29*, 156–163, 2001.

[13] Punakivi M., "Comparing Alternative Home Delivery Models for E-Grocery Business", Doctoral Dissertation, Department of Industrial Engineering and Management, Helsinki University of Technology, Finland, 2003.

[14] Rousseau L.-M., Gendreau M., Pesant G., "Using Constraint-Based Operators to Solve the Vehicle Routing Problem with Time Windows", *Journal of Heuristics 8*, 43–58, 2002.

[15] Solomon M.M., "Algorithms for the Vehicle Routing and Scheduling Problem with Time Window Constraints", *Operations Research 35*, 254–265, 1987.

[16] Taillard É. D., G. Laporte, M. Gendreau, "Vehicle Routeing with Multiple Use of Vehicles", *Journal of the Operational Research Society 47*, 1065-1070, 1996.

[17] Yrjölä H., "Physical Distribution Considerations for Electronic Grocery Shopping", *International Journal of Physical Distribution and Logistics Management 31*, 746–761, 2001.